

UPPSALA UNIVERSITY

DECENTRALIZED AIR QUALITY MONITORING AND
PREDICTION

(1DT054) - AUTUMN 2020

Product Report

Group 2:

Dhanush Kumar AKUNURI
Md Tahseen ANAM
Jude FELIX
Lokesh KUMAR
Venkata Sai Teja MOGILLAPALLE
Seema NEGI
Anil POUDEL
Ankit PANDEY
Vandita SINGH
Praveen SWAMY

January 17, 2021

Abstract

This report presents an application of Deep Learning techniques to predict air pollution with time series data. The product is called Decentralized Air Quality Monitoring and Prediction(DAMP). Air Quality monitoring has become an important part of living a healthy life. Monitoring air quality helps in assessing the level of pollution in relation to the ideal air quality standards. In this project, the values of four pollutants namely PM10, NO2, NOX as NO2 and PM2.5 is predicted in the Stockholm region. The prediction is done using a centralized model and a federated model and the performance is compared. Further, for centralized model, a Long Short Term Memory(LSTM) model and a Deep Neural Network(DNN) model are used and the performance is further evaluated and compared. The predictions are done for 24 hours, 6 hours and 1 hour. Prior to training the network, the dataset was pre-processed and missing values were imputed using KNN imputer. Finally for evaluation, Mean Absolute Error(MAE) and Symmetric Mean Absolute Percentage(SMAPE) Error were used. To visualise the predictions, a DAMP WebApp is created, which helps the users understand the current Air quality values. This paper is a great motivation to continue the research on how to monitor air quality.

Contents

1	Introduction	1
1.1	Need for Air Quality Monitoring	1
1.2	De-centralized Air Quality Monitoring and Prediction (DAMP)	1
1.3	Report Outline	2
2	Preliminaries	3
2.1	Data set	3
2.2	Data Visualization Methods	3
2.3	Pre-processing Methods	9
2.4	Centralized Model for Time-Series Forecasting	12
2.5	Federated Learning	17
2.6	Evaluation Metrics	18
3	Related Work	19
4	Methodology	22
4.1	Data Collection	23
4.2	Implementation Details	23
4.2.1	Data Visualization and Pre-processing	23
4.2.2	Centralized Model - Development and Optimization . .	28
4.2.3	Federated Learning Model	31
5	Results and Discussion	34
6	Decentralised Air Quality Prediction and Monitoring (DAMP)	
	- The Product	40
6.1	DAMP Web Application	40
6.1.1	Technologies and Libraries	42
7	Conclusions	44
8	Future Work	46
A	Appendix A: Instructions to use the Web Application	I
A.1	Step I	I
A.2	Step II	I
A.3	Step III	I

A.4 Step IV	I
-----------------------	---

1 Introduction

1.1 Need for Air Quality Monitoring

With the advent of industrialisation, pollution in various forms- Land, Water and Air has been growing. One of the major concerns is the Air Pollution which negatively affects not only the environment but also the lives of human-beings living in urban and rural areas in almost every part of the world. The quality of air we breathe largely impacts the quality of life of people from all age-groups, while the hazardous effects of this type of pollution could be extremely dangerous to those falling under certain sensitive groups.

Air Pollution could be caused by natural factors such as gases released from volcanoes, smoke from bush-fires as well as man-made factors like factories, motor-vehicles, coal based electricity generation plants mostly resulting into releasing considerably large concentrations of harmful gases in the air. When bad quality air is breathed in by healthy humans on a regular basis, it may result into developing from slight discomfort to long-term ailments such as heart problems, premature birth, chronic bronchitis, pneumonia, and emphysema[1]. However, in case of risk-groups, the effect of breathing in bad quality air may be aggravated to fatal health conditions such as cancer, asthma etc.

1.2 De-centralized Air Quality Monitoring and Prediction (DAMP)

The objective was to develop a product for monitoring and prediction of Air Quality in Stockholm by using Federated Learning. The system takes as input air concentration data along with selective meteorological parameters to predict the Air Quality at different stations in Stockholm. The major pollutants of concern were Nitrogen dioxide (NO_2) and Particulate Matter (PM_{10}).

The data was provided by Sweden's Meteorological and Hydrological Institute (SMHI). The product is targeted to provide a forecast stating the level of pollution in Stockholm which would be helping people belonging to certain sensitive groups in deciding when they should/should not go out and thus reduce their risk of being exposed to Air Pollution.

1.3 Report Outline

This report comprises of seven sections in all. The objective and the motivation has been introduced in Section 1. Section 2 provides a brief insight into the concepts taken into consideration for the overall development of the product at different stages. Here, all the techniques that were applied to build different variants of the initial product have been discussed. The Section 3 presents a comprehensive survey of techniques deployed for Decentralized Air Quality Monitoring and Prediction for various Air Quality data sets. The system architecture and the implementation details have been discussed in the Section 4. Section 5 provides an overview of the results and plots obtained at different stages of the development of the system. Section 6 presents the product from the user's perspective, mostly discussing about the front end and the visualization of predictions in user understandable form. The possible inferences and conclusions drawn from the results, limitations, challenges and possibilities for future work have been presented in the Section 7.

2 Preliminaries

2.1 Data set

The dataset used in this project consisted of air pollution data (PM10, PM2.5, NO2, NOX as NO2) captured from 4 monitoring stations and meteorological data (temperature, pressure, global radiation, relative humidity, wind speed, wind direction) obtained from 1 weather station between 2015 and 2019. The dataset was obtained from SMHI.

Selecting the below options resulted in data from 13 air quality stations. Län = Stockholms Län, Kommun = Stockholm, År = 2015 to 2019, Tid-supplösning = Hour, Ämne = Alla, Station = Alla, Klassificering = Background and Traffic.

Only four stations shown in Table 1 had data available for the given time period and the common air pollutants that are mentioned above.

Station Id	Station Name
18644	Stockholm E4/E20 Lilla Essingen
8799	Stockholm Sveavägen 59 Gata
8780	Stockholm Hornsgatan 108 Gata
8781	Stockholm Torkel Knutssonsgatan

Table 1: Air Quality Stations for the data

2.2 Data Visualization Methods

Libraries

- **Seaborn:** Seaborn is a python library based on matplotlib and used for data visualization[2]. We have used this library to generate heatmap for correlation matrix, violinplot and box plots for outlier detection.
- **Missingno:** Missingno provides a small toolset to visualize missing data or completeness of the dataset in an easy way[3].

-
- **Matplotlib:** Matplotlib is a plotting library used in this project to generate scatter plots, line plots for different features[4].

Methods

- **Heatmap:** Seaborn's Heatmap method was used to visualize the correlation matrix[5]. The correlation matrix shows the correlation between different features in the dataset. Seaborn generates heatmap of correlation matrix using Pearson's correlation. Figure 1 is an example of the heatmap of a correlation matrix. The correlation value ranges from -1 to 1 where the value 1 indicates that the corresponding features are perfectly correlated and 0 indicates that they are not correlated and negative value represents negative correlation. If a cell has bright color then it means that the features containing that cell are strongly correlated and if a cell has dark color then it means that the features containing that cell are poorly correlated. The example figure shows correlation matrix of breast cancer data where the features are represented along the x axis and y axis.

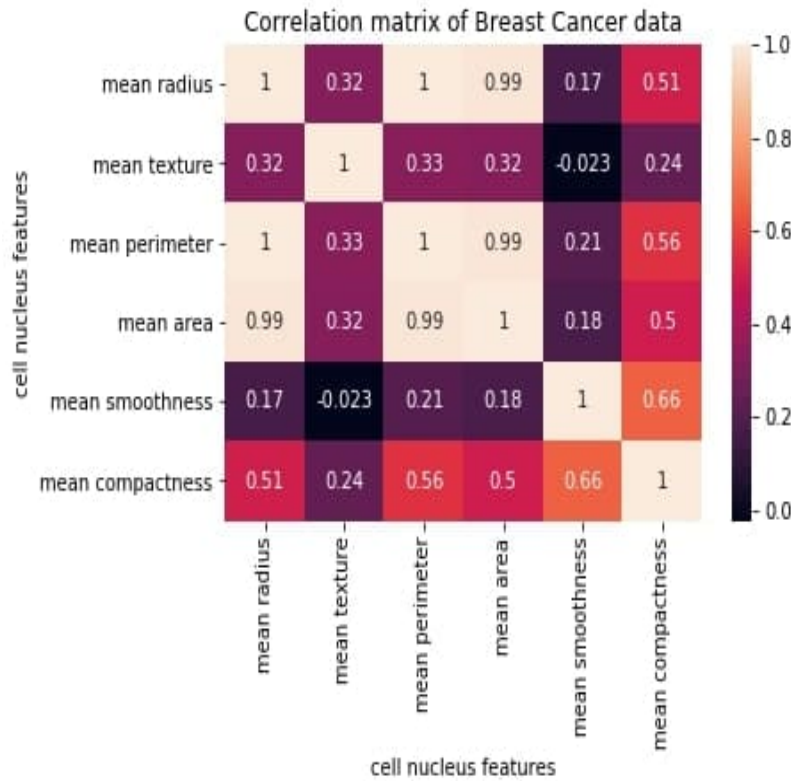


Figure 1: Correlation Matrix[6]

- **Missingno:** Missingno provides tools which were used to generate plots to show the completeness of our dataset. Figure 2 is an example of the missingno plot showing missing values in different features. The white spots indicates missing values. As we can see in the figure below, some features like contribution factor of the first vehicle appear to be completely populated while some appears to be mostly populated and some are not populated at all.

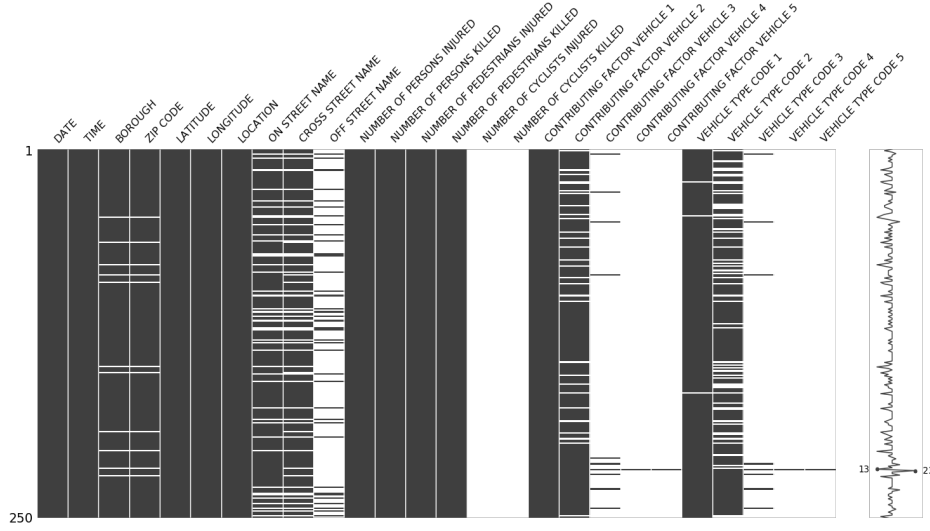


Figure 2: Missingno plot[7]

- Violinplots:** Violinplots were generated to get a visual representation of the probability of outliers present in the features of our dataset[2]. Violin plot draws a combination of boxplot and kernel density estimate[8]. Kernel Density Estimations represents the shape/distribution of the data[9]. The figure 3 shows an example of violin plot. In a violin plot, wider sections represent a higher probability of observations taking a given value while the thinner sections represent lower probability of observations[10]. Therefore, observations lying outside of the first quartile and third quartile of the boxplot can be considered as outliers[10].

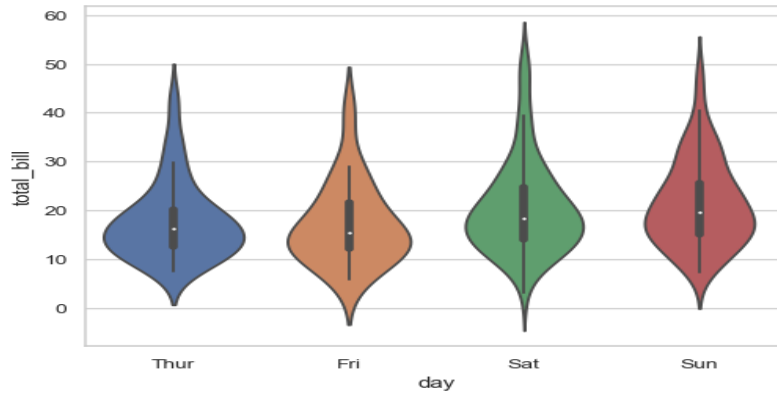


Figure 3: Violin plot[8]

- **ScatterPlots:** Scatterplots were also used in this project during outlier detection[11]. An example is shown in the figure 4. The example in the figure shows a scatter plot between life expectancy values (lifeExp) and gdp per capita (gdpPerCap) values. There are few data points with large gdpPerCap values stand out as outliers[12].

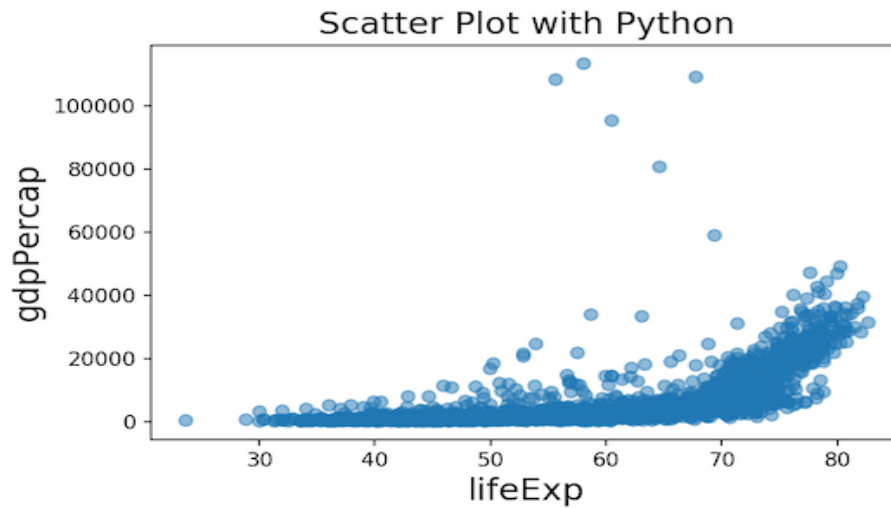


Figure 4: Scatter plot[12]

- **PairPlots:** Pairplots were used to get the visualization of the relation-

ship between different features in our dataset[2]. Pairplots gave us a more visual representation of how the features are related to each other while correlation matrix showed us how strong or weak the relationship is between features. Figure 5 shows a simple pairplot that uses scatterplot for each pairing of the variables and histplot for the marginal plots along the diagonal[13].

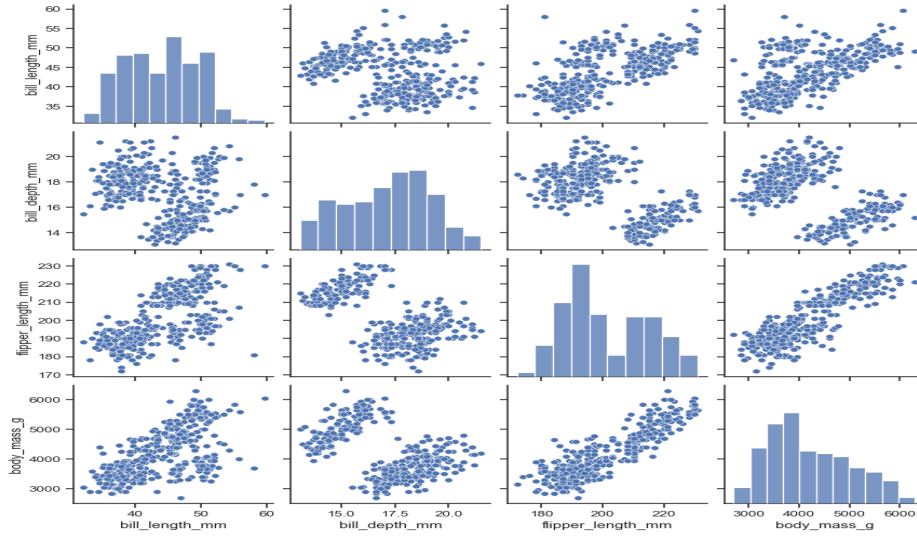


Figure 5: Auto Correlation plot[13]

- **Auto Correlation Plots:** Auto correlation plots were generated to observe how the present value of a feature is correlated with its past values. It is a bar chart coefficients of correlation between a time series and its lagged values[14]. Figure 6 shows an example of a simple autocorrelation plot where the x-axis represents the lag and the y-axis represents the correlation. Each spike that rises above or falls below the dashed lines is considered to be statistically significant[15]. If a spike is significantly different from zero, then that is an indication of autocorrelation and if not, then it can be considered as evidence against autocorrelation[15].

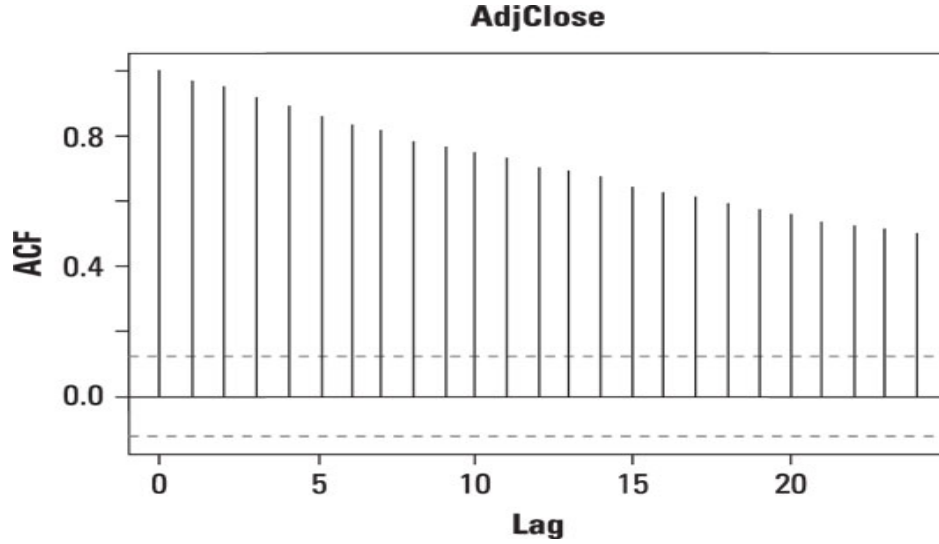


Figure 6: Auto Correlation plot[15]

2.3 Pre-processing Methods

In real world the data collected is either incomplete, noisy or inconsistent. Data Pre-processing is a technique which transforms raw data into meaningful form. The performance of the model is heavily dependent on data which is it trained on. Training a model with a dataset which is pre-processed can drastically impact the model's accuracy.

Imputation Methods

Imputation Methods are used to compensate the missing values of the data set with the interpolated values utilizing non-missing values of the data set. Some of these Techniques employed during the work are described as below:

- **Mean/Median:** Mean/Median technique can be used to fill the missing values of a column by calculating the Mean/Median value of the non-missing value of a column. Target column must contain the numeric type data.

	col1	col2	col3	col4	col5			col1	col2	col3	col4	col5	
0	2	5.0	3.0	6	NaN	mean()		0	2.0	5.0	3.0	6.0	7.0
1	9	NaN	9.0	0	7.0			1	9.0	11.0	9.0	0.0	7.0
2	19	17.0	NaN	9	NaN			2	19.0	17.0	6.0	9.0	7.0

Figure 7: Mean/Median Imputation

- **Zero or Constant Values:** Under this technique we replace the missing values either by zero or by a constant value.
- **Forward and Backward filling:** Forward filling and backward filling are two approaches for filling missing values in the data. In Forward filling missing values are replaced with previous data point. In Backward Filling, missing values are replaced with next data point.
- **Linear Interpolation:** Linear Interpolation is a curve fitting method used to fill missing values between a set of discrete data points. This method fits a straight line between previously observed and next observed values[16].The line equation is given as

$$y = y_1 + [(y_2 - y_1)/(x_2 - x_1)](x - x_1). \quad (1)$$

Where y is interpolant, x is time of interpolant, (x_1, y_1) are coordinates of starting points of gap and (x_2, y_2) are end point coordinates.

- **Multiple Imputation by Chained Equation(MICE):** MICE is a multiple imputation method used to replace missing values in a data set under certain assumptions about missingness mechanism (the data is missing at random)[17].

MICE operates under an assumption that the data which is to be imputed has missing values at random[18]. MICE when implemented on non-random missing values would result in biased estimates. In MICE procedure a series of regression models are run whereby each variable with missing data is modeled conditionally upon the other variables of the data. For example if the distribution consists of binary variables logistic regression is used and in case of continuous variables linear regression is used for modeling. Software packages do vary somewhat in

their implementation of MICE, with some packages also using a multinomial logit model for categorical variables and a Poisson model for count variables[18].

- **K-Nearest Neighbours (K-NN):** This technique employs k-Nearest Neighbour classification algorithm which works on the principle of similarity. Each column is clustered into k different groups/clusters and the missing value is then calculated based on the identification of the closest cluster of similarity.

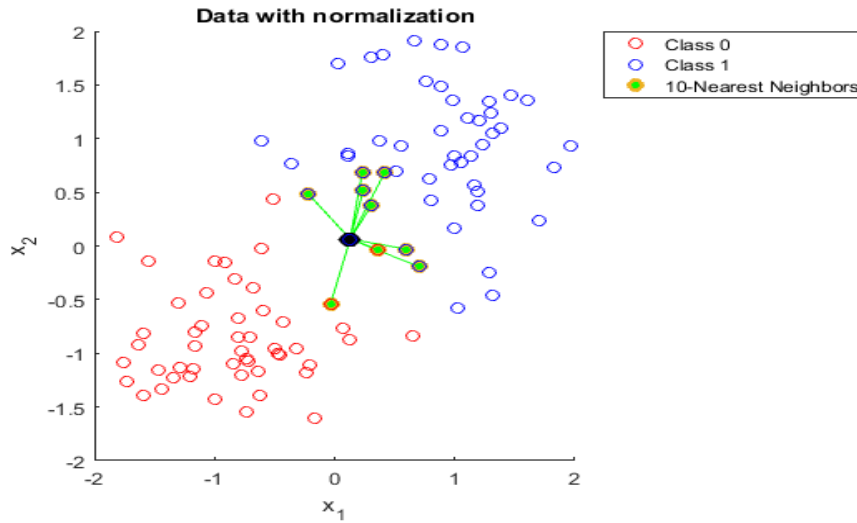


Figure 8: k-Nearest Neighbour Imputation

Outlier Detection Methods

Outlier Detection Methods are used to detect and eliminate the extreme values from the data set. Some of these techniques which were investigated are described as followed.

- **Computing Z-Score:** Z-scores can be used to find the outlier when the data follow the normal distribution. It indicates how many standard deviations away a given observation is from the mean. The formula for the Z-Score is given by:

$$Z = (x - \mu) / \sigma$$

where Z = Z-score

x = observed value

μ = mean of the sample

σ = standard deviation of the sample

- **Isolation Forest Method:** Isolation Forest algorithm employs the fact that anomalous data points are few and significantly different from ‘normal’ data points. Isolation Forest uses the decision tree algorithm. To isolate the anomalous data points, it randomly selects a feature from the given set of features. It then randomly selects a split value between that feature’s max and min values.
- **MCD:** The minimum covariance determinant method is a highly robust estimator of multivariate location and scatter. Since estimating the covariance matrix is the cornerstone of many multivariate statistical methods, the MCD is an important building block when developing robust multivariate techniques. It also serves as a convenient and efficient tool for outlier detection.[19]
- **Local Outlier Factor (LOF):** LOF uses the density of data points in the distribution as a key factor to detect outliers. It finds the outliers by comparing the density of a given data point to its neighbours’ density. The higher the value of LOF, the more likely it is an outlier.

2.4 Centralized Model for Time-Series Forecasting

Time series forecasting is predicting the values in future using the historical data from the past. Time series data are collection of observations made sequentially through time[20]. There are two different forecasting methods,

- **Univariate method:** It is a method that uses the present and past values of a single series to be forecasted.
- **Multivariate method:** Forecasts of a given variable are depending on one or more time series variable.

Machine learning models have given more improved performance and accuracy[21] when compared to the classical statistical models. Along with the intention to perform federated models, the deep learning models are used for forecasting.

Deep neural network(DNN)

Deep neural network(DNN) is a subset of the machine learning and artificial intelligence. DNNs are an extension of the artificial neural networks which has multiple hidden layers to solve complex functions[22]. Deep neural networks have a input layer, hidden layers and a output layer. All these layers have the same components: neurons, synapses, weights, biases, and functions[23]. The figure 9 shows a simple architecture of a DNN with a input layer, three hidden layers and a output layer.

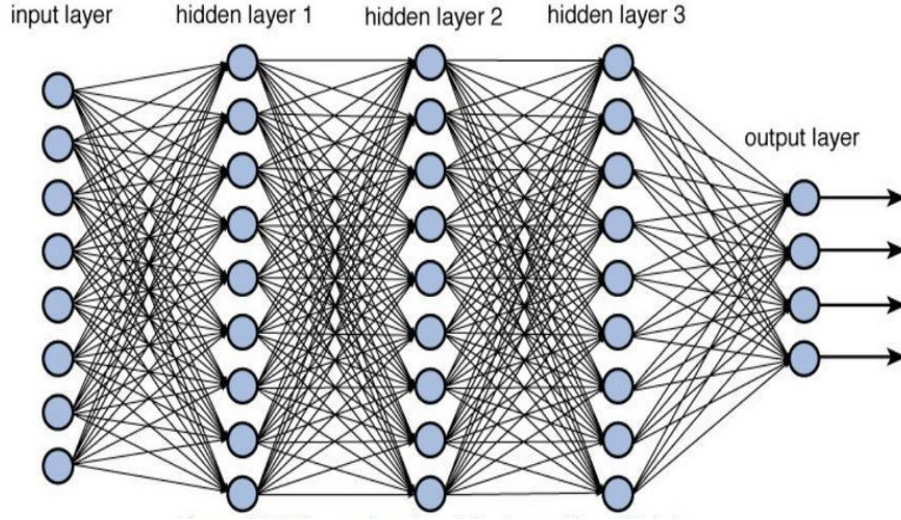


Figure 9: Simple architecture of Deep neural network

The DNNs contain many simple *neurons* producing a sequence of real-valued activation. The neurons get activated in the input layer through the inputs fed and the other neurons in the hidden layers get activated through the weighted connections from the previously active neurons[23]. This process can be seen in the figure 9.

A *synapse* is the connection between two neurons. Each synapse has a *weight* associated to it. The neurons with higher weights will be dominant

over the neurons with the lower weights. A *bias* neuron is added to all the neurons along with the weights to allow a broader storage and add richer representations of the input space [24].

Recurrent neural networks(RNN)

A recurrent neural network(RNN) is a class of artificial neural networks that are naturally suited to process time-series data and other sequential data. RNNs are designed for analyzing streams of data by means of hidden units.

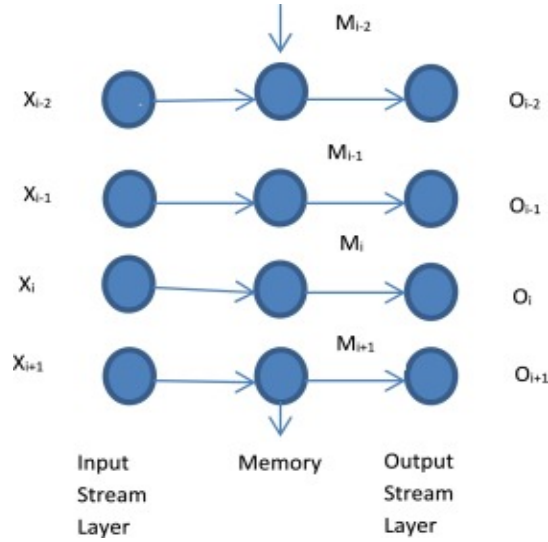


Figure 10: RNN

RNNs are provided with the input samples which contain more inter-dependencies[25]. The output produced at time t_1 affects the parameter available at time t_1+1 [25]. In this way, RNNs keep two kinds of input such as the present one and the past recent one to produce the output for the new data[25].

A transition function is applied to all of its hidden states in a recursive manner. The activation function followed by the RNNs hidden state is a function that depends on its previous states only[25]. The sequence of inputs are mapped into a fixed size vector and then it is fed as an input to a softmax activation function to produce the output.

Regular RNNs might have a difficulty in learning long range dependencies. During the back propagation stage of training process, if any of the gradients

equate to 0 then all the gradients would equate to zero due to multiplication. Such states would no longer help the network to learn anything. This is known as Vanishing gradient problem[25].

Apart from this problem RNNs also become very difficult to train due to extremely large number of parameters while handling long term dependencies. Thus, a special kind of RNN - Long Short Term Memory networks (LSTM), that are explicitly designed in learning long-term dependencies[26].

Long Short Term Memory networks(LSTMs)

LSTMs also have a chain like structure similar to RNNs, but the repeating module has a different structure[26]. Instead of having a single neural network layer, there are four, interacting in a very special way[26].

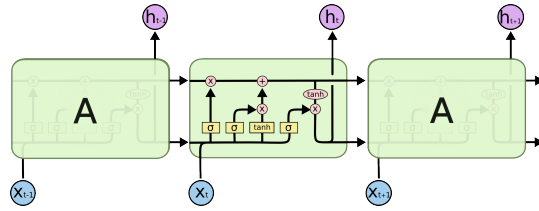


Figure 11: LSTM

The key to LSTMs is the cell state, which is kind of like a conveyor belt. It runs straight down the entire chain, with only some minor linear interactions. The LSTM does have the ability to remove or add information to the cell state, carefully regulated by structures called gates. They are composed out of a sigmoid neural net layer and a point wise multiplication operation. The sigmoid layer outputs numbers between zero and one, describing how much of each component should be let through. An LSTM has three such gates, to protect and control the cell state.[26]

- Forget gate - is responsible for removing the information that is no longer required or that is of less importance for the LSTM from the cell state via multiplication of a filter.

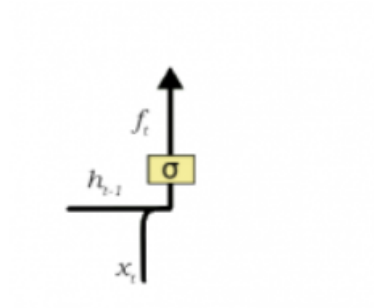


Figure 12: Forget gate

- Input gate - is responsible for adding new information to the cell state.

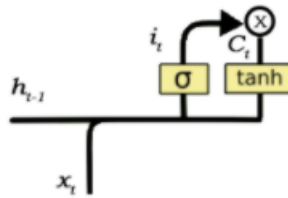


Figure 13: Input gate

- Output gate - is responsible for selecting useful information from the current cell state and showing it out as an output.

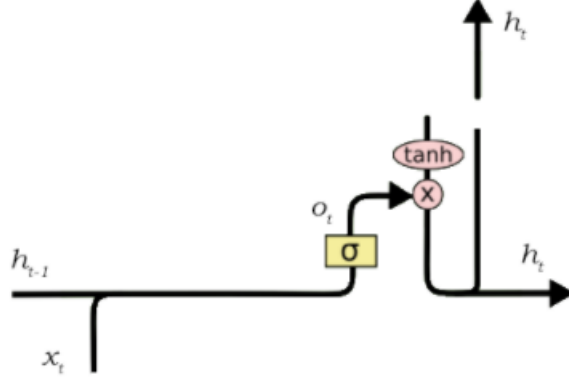


Figure 14: Output gate

Attention Mechanism

Attention is a mechanism targeted to compute the alignment score between two sources by means of a compatibility function which aims to measure the dependency between these two sources.[27]. The softmax function is responsible for transforming these scores into probability distribution by normalising over all given iterations of the first source entity. Hence larger distribution would mean that the given token of the source contributes more important information to the second given entity. An example would be how much contribution a given feature makes for a given time step.

2.5 Federated Learning

Mobile phones, wearable devices and autonomous vehicles are just a few of the modern distributed networks generating a wealth of data each day[28]. Due to the growing computational power of these devices - coupled with concerns over transmitting private information - it is increasingly attractive to store data locally and push network computation to the edge[28].

Recent works considered training machine learning models centrally but storing and serving them locally. However, as the storage and computational capabilities of the devices within distributed networks grow, it is possible to leverage enhanced local resources on each device.[28] This has led to a growing interest in federated learning FL, which explores training statistical

models directly on remote devices. Federated learning methods play a critical role in supporting privacy-sensitive applications where the training data are distributed at the edge.[28]

In a typical machine learning system, an optimization algorithm like Stochastic Gradient Descent (SGD) runs on a large dataset partitioned homogeneously across servers in the cloud. Such highly iterative algorithms require low-latency, high-throughput connections to the training data.[29] But in the Federated Learning setting, the data is distributed across millions of devices in a highly uneven fashion. In addition, these devices have significantly higher-latency, lower-throughput connections and are only intermittently available for training.[29]

2.6 Evaluation Metrics

Mean Absolute Error(MAE)

Mean absolute error (MAE) is a measure of errors between paired observations expressing the same phenomenon.[30] Examples of Y versus X include comparisons of predicted versus observed, subsequent time versus initial time, and one technique of measurement versus an alternative technique of measurement.[30] MAE is calculated as:

$$MAE = \frac{\sum_{i=1}^n |y_i x_i|}{n} = \frac{\sum_{i=1}^n |e_i|}{n} \quad (2)$$

It is thus an arithmetic average of the absolute errors $|e_i| = |y_i - x_i|$, where y_i is the prediction and x_i is the true value.

Symmetric Mean Absolute Percentage Error(SMAPE)

Symmetric mean absolute percentage error (SMAPE) is an accuracy measure based on percentage (or relative) errors.[31] It is usually defined as follows:

$$SMAPE = \frac{1}{n} \sum_{t=1}^n \frac{|F_t - A_t|}{(A_t + F_t)/2} \quad (3)$$

In contrast to the mean absolute percentage error, SMAPE has both a lower bound and an upper bound.[31]

3 Related Work

Air Quality Prediction

There was a collaborative research project that focuses on the findings on which air quality is measured, how those findings affect the quality of the air and ongoing models to analyse the data of air quality measurements based on the study of the air in a major area in Mexico for a certain period of time.[32]

Another article focuses on the challenges for air quality prediction and proposes a design paradigm to overcome those challenges. At the end of this article, there are several evaluation techniques that can be taken under consideration to evaluate the correctness of the model. [33] Recurrent Neural Network has been suggested to overcome the challenge of temporal dependency and Convolutional Neural Networks (CNNs) are adopted to capture spatial correlation. Also it has been suggested to address these challenges from both feature and model perspective.[33]

Another work predicts air quality using deep learning. 8 hours averaged surface ozone(O₃) concentrations were predicted using deep learning consisting of a recurrent neural network (RNN) with long short-term memory (LSTM), to accurately predict local 8-hr averaged (ave) O₃ concentrations based on hourly air monitoring station measurements as a tool to forecast air pollution. RNNs are particularly well suited for air concentration prediction because they incorporate sequential history into the training and processing of input data. Air quality measurements are time-series data sets in which the order of data is important.[34]

In [35], the authors focus on outlier detection, targeting to detect air pollution events as well as removing errors that could affect the data analysis and comparison at later stages. The objective of this study was to develop an adequate outlier detection method for an urban air quality sensor network, characterized by a fine-scale spatial and temporal variation in air quality. The author has based the study in the city of Eindhoven, from an air quality sensor network for NO₂ data. The method proved useful for distinguishing outliers in an area with high spatial and temporal variability of air pollutant concentrations, providing a basis for future work on differentiating between types of outliers- errors and events.

In this project, we have tried to predict air quality across different stations using a centralized model and implemented federated learning to improve the performance of the model. We have considered two groups of features, air quality and meteorological data to better characterize the interactions among all the influencing factors.

The authors in [36] propose a winning solution to Air Quality Prediction which focuses on the challenges for air quality prediction and proposes a design paradigm to overcome those challenges. At the end of this article, there are several evaluation techniques that can be taken under consideration to evaluate the correctness of the model. The development was done using an ensemble system as the following constituents - LightGBM, GatedDNN and Seq2Seq, where LightGBM could be a feature selector which works good with noisy data as well. Gated-DNN was useful for learning salient representations using deep NNs, thus capable of embedding complex spatio-temporal relatedness into different dimensions. Seq2Seq was said to inherit the encoder-decoder structure.

Attention Mechanism was introduced in deep networks as encoder-decoder structure by Bahdanau et. al. [37]. Shen et. al.[27] introduced directional self-attention networks for Natural Language Processing based tasks. There have been many variants for the Attention Mechanism such as Sequence Self Attention. The encoder is mostly RNN which transforming the input sequences into feature based representation. In order to apply the Attention Mechanism for sequential or temporal data, the mapping from input to the hidden state can be made to learn at each time step.

With the main idea to build machine-learning models based on distributed datasets, Google proposed the concept of Federated learning in 2016. There have been several efforts to make Federated Learning more secure and personalized. Further, the authors in [38] present the definition of Federated Learning, approaches for implementing the privacy property of Federated Learning, categorization of Federated Learning, architecture of the federated learning model and the applications of Federated Learning.

Approaches for privacy preservation identified by the authors in [38] were (i) Secure Multiparty Computation (SMC) involving multiple parties and providing security proof such that each party knows nothing except its input and output ; (ii) Differential Privacy, also known as k-anonymity for data privacy protection which involves adding noise to the data or generalization,

(iii) Homomorphic Encryption for centralizing and training data on the cloud. The potential challenge for privacy-preservation is the Indirect Information Leakage. This happens when intermediate results such as parameter updates in form of gradients are leaked, there may happen the leakage of important data information too.

Based on the distribution characteristics of data, Federated Learning can be categorized as follows - Horizontal Federated Learning, Vertical Federated Learning and Federated Transfer Learning [38]. For each of these categories, the authors have illustrated the general architectures. According to the authors, Federated Learning could be considered as “privacy-preserving, decentralized collaborative machine learning” which could be useful for implementation of privacy-preserving protocols for logistic regression and neural networks.

In addition, the authors in [38] focus on the following key-points: Federated learning being different from distributed machine learning since FL faces more complex learning environment and emphasizes the data-privacy protection. Federated Learning being seen as an operating system for edge computing. Federated learning becomes different from Federated databases since Federated databases do not employ privacy protection mechanism and only focuses in the basic operations of data-insert, delete, search, query.

4 Methodology

The overall design of the system has been built in four phases, first being the Data Collection phase followed by Data Visualization and Pre-processing Phase. Once the data was pre-processed and cleaned, some additional time-based features were extracted. The Development and Optimization of the Centralized Model was done to predict the air quality across different stations, which was then followed by the Implementation of the Federated Model. The algorithmic flowchart could be described in the Figure 15.

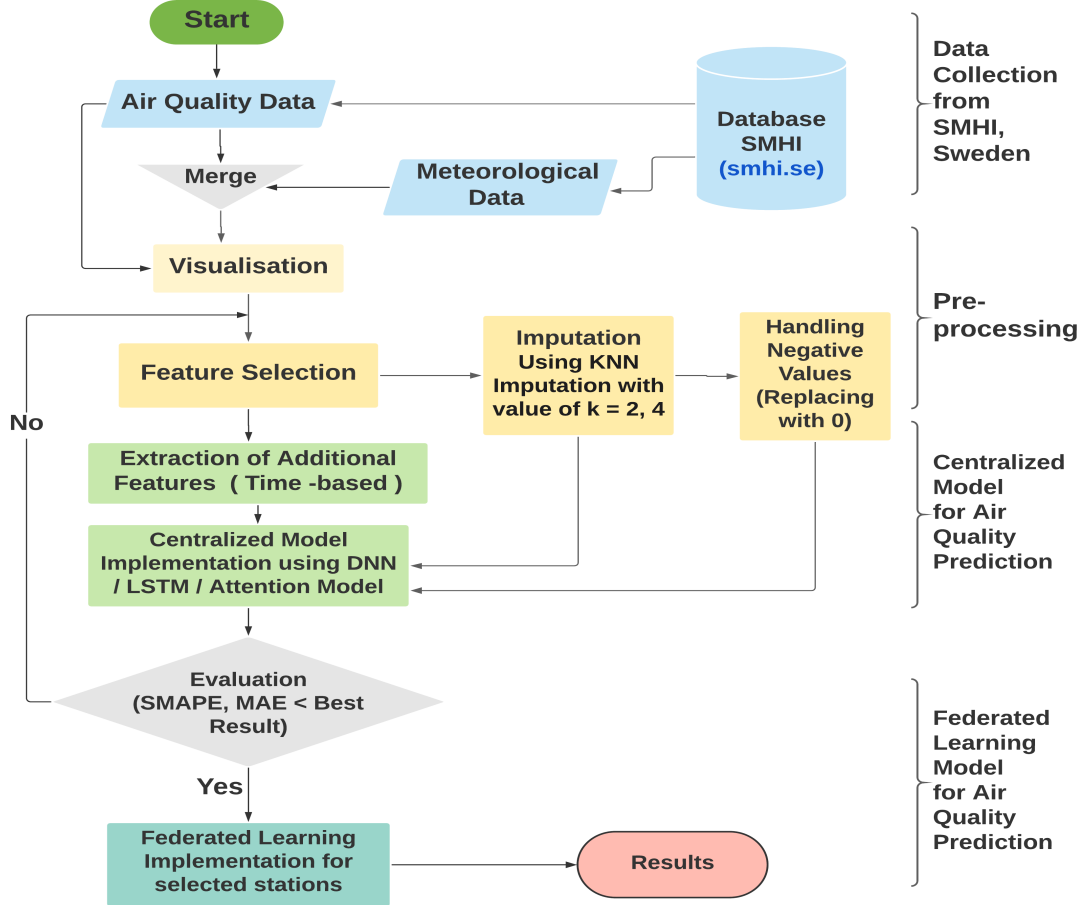


Figure 15: Work Flow Diagram for DAMP

4.1 Data Collection

This was the first phase of the system development. During this phase, the data was collected from SMHI website [39], [40] and the various stations were analysed for the given air concentration and meteorological features. Initially all the stations were analysed for the available features. Table 2 shows the final selected stations from the SMHI dataset that was provided. These stations were selected on the basis of the availability of the pollutant *PM10* for the year 2014 to 2019. The other pollutants like *NO2*, *NOX* as *NO2* and *PM2.5* were present in all the selected stations and hence the final dataset had these four pollutants. The meteorological features like temperature, relative humidity, precipitation, wind velocity(speed and direction) and air pressure were having impact on the concentrations of certain air elements [41].

Time becomes a very important feature in forecasting problems. The system here has to deal with forecasting 24 hours of weather features into the future. Therefore, the hour of the day and day of the week were incorporated into the data set along with the air concentrations and meteorological features. The models were designed to forecast the four pollutants *PM10*, *PM2.5*, *NO2* and *NOX* as *NO2* for 24 hours with a input of 24 hours.

4.2 Implementation Details

4.2.1 Data Visualization and Pre-processing

Visualisation

Correlation plots as explained in section 2.2 are used to find the correlation between the features that was obtained from SMHI dataset. The figure 16 shows the correlation matrix of the station *Stockholm Hornsgatan 108 Gata*.The correlation matrix was calculated using the Pearson’s correlation. All the features had low correlation to *PM10* pollutant. The meteorological features had very low and but some showed negative correlations. This could be because there was only one meteorological station recorded for the whole of Stockholm city and the air concentration stations were plenty. The selected air concentration features had better correlation values when compared to the meteorological features. The air temperature and precipitation were considered as they were better correlation values when compared to the other meteorological features.

The pairplots were plotted to visualize the relationships between the fea-

Station	ID	2014	2015	2016	2017	2018	2019
Stockholm E4/E20 Lilla Es- singen	18644	NO2, NOX as NO2, PM10, PM2.5	NO2, NOX as NO2, PM10, PM2.5	NO2, NOx, PM10, PM2.5	NO2, NOX, PM10, PM2.5	NO2, NOX, PM10, PM2.5	NO2, NOX as NO2, PM10, PM2.5
Stockholm Sveavägen 59 Gata	8779	NO2, NOX as NO2, PM10, PM2.5, CO, O3	CO, NO2, NOX as NO2, PM10, PM2.5	NO2, NOX, PM10, PM2.5	CO, NO2, NOX,	PM10, PM2.5, CO, NO2, NOX, PM10, PM2.5	CO, NO2, NOX as NO2, PM10, PM2.5
Stockholm Horns- gatan 108 Gata	8780	NO2, NOX as NO2, PM10, PM2.5, CO	Black Car- bon, CO, NO2, NOX as NO2, O3, PM10, PM2.5	NO2, NOX as NO2, PM10, PM2.5	ALL	ALL	Black Car- bon, CO, NO2, NOX as NO2, PM10, PM2.5
Stockholm Torkel Knutssons- gatan	8781	NO2, NOX as NO2, PM10, PM2.5, O3	Black Car- bon, NO2, NOX as NO2, O3, PM10, PM2.5	NO2, NOx, PM10, PM2.5	CO Miss- ing, ALL	CO Miss- ing, ALL	Black Car- bon, NO2, NOX as NO2, O3, PM10, PM2.5

Table 2: Data Collection: Selected Stations

tures. The figure 17 shows the relationships between the features from the station *Stockholm Hornsgatan 108 Gata*. Similar to the correlation matrix, the pair plots do not reveal any major relationships or correlation between the features.

Missingno plots was used to find if there were any missing data when collected from the SMHI dataset. All the stations had missing values among

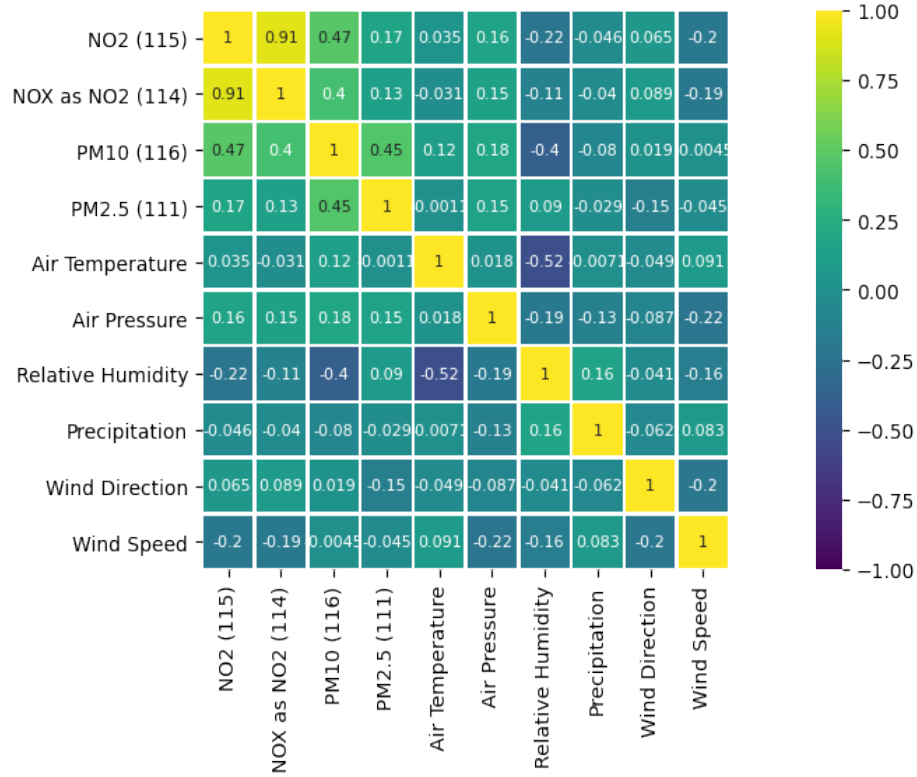


Figure 16: Correlation matrix of the features from the station 8780.

all the four air pollutants in the range of 2-10%. The figure 18 shows the missing values of the Stockholm Hornsgatan 108 Gata station. The white lines in the vertical bars reflect the missing values in the dataset.

Imputation Techniques

The missing data had to be filled in with some values in-order to have better performance of the model. The forecasting problems need to the data to be sequential and hence the dropping the missing values cannot be done and filling in with a constant values like 0 or mean or median may not yield satisfying results. The section 2.3 talks about several imputation techniques that was carried out for the data. The values generated for the missing values cannot be validated, the values generated from different techniques were fed into a simple model to see which data performs better. The technique using

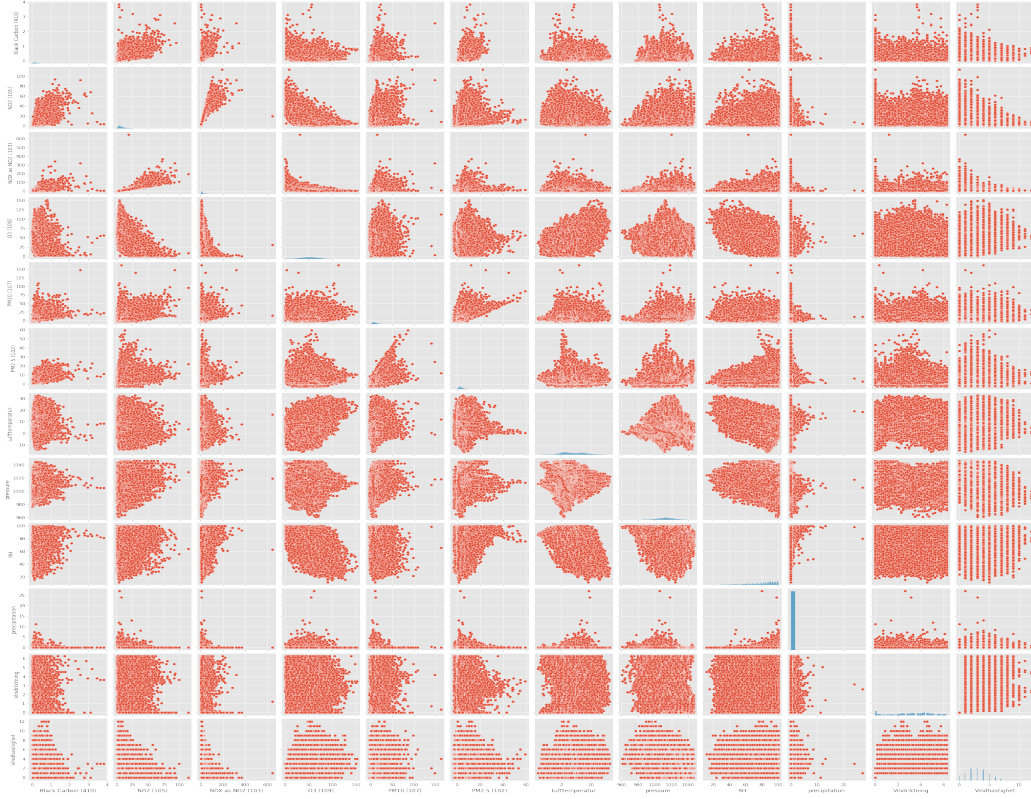


Figure 17: Pair plot of the features from the dataset.

the k-nearest neighbours algorithm had better results when compared to the other techniques. Hence, the k-nearest neighbours algorithm technique was being used for the missing values.

The missing values being filled, there is always a chance of the certain values in the dataset to be an outlier. This could be because of the instruments failure or some mistakes that happen during the recording of the values. The violin plots and scatter plots are certain plotting techniques used to visualize the distribution of the features and find the outliers if present. The figures 19 and 20 shows the violin and scatter plot of the Stockholm Hornsgatan 108 Gata station. There are several techniques that was implemented as explained in the section 2.3. The results were not satisfying as the performance of those datasets were poor. The air pollutants cannot have negative values as they are the concentration values of the pollutant in the air present near the station. Hence, the negative values were converted into missing values

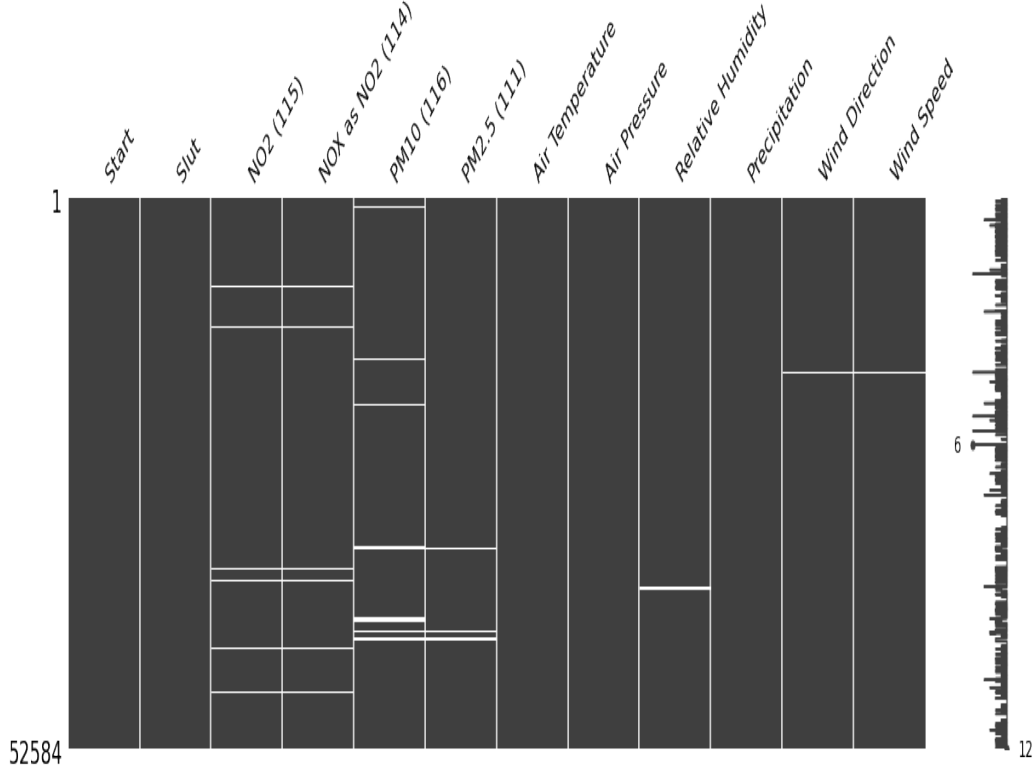


Figure 18: Missingno plot for the station 8780.

and then imputed with k-nearest neighbour technique.

The centralised models based on DNN and LSTM were developed using K-Nearest Neighbour Imputation technique with value of K as 4.

Features Description

The different features have values in varied ranges. The air pollutant concentrations ranges are between 0-800 and the values of the meteorological features such Temperature are negative values. Some of the meteorological features used were Air pressure, Temperature and Global Radiation.

All the features need to be normalised before using it on the models. The standardization method was used to normalize the data. On standardizing the data they fall in the range -1 to +1.

Certain time-based features were extracted which proved useful in the

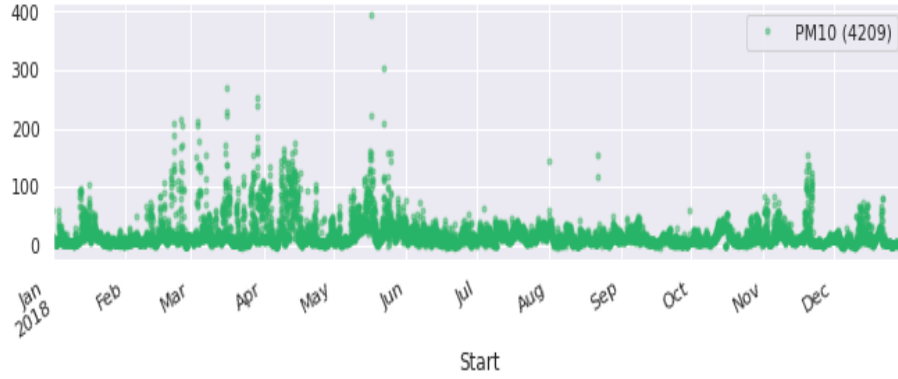


Figure 19: Scatter plot of the PM10 values.

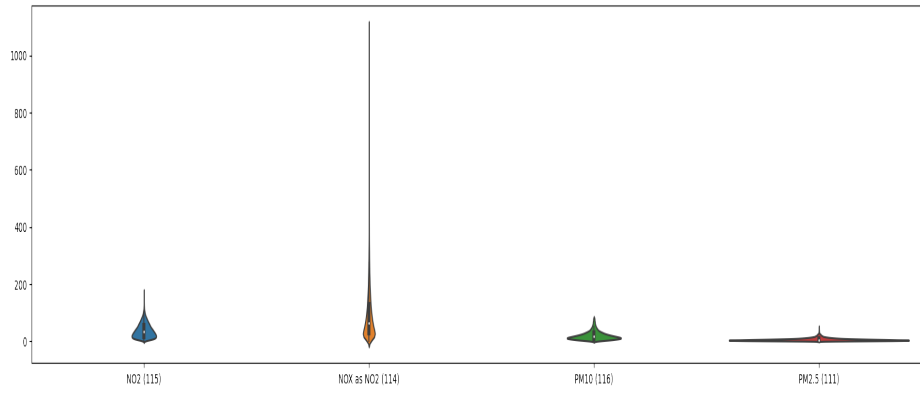


Figure 20: Violin plot for the station 8780.

later stages of the development of the model. Features such as Weekday, Time of the day, Week number, day sin, day cos, year sin and year cos (To form the day and year as signal as to understand the importance of the feature at different frequencies) were extracted and used for the centralised model.

4.2.2 Centralized Model - Development and Optimization

Models for centralized data is called as centralized model. Centralized data are the data that are gathered and stored at a single place. The four station data were accessed from the SMHI database and were appended together

for the centralized model. There are two models used for the centralized approach for the air quality : *Deep neural network and Long short term networks*.

The data set was split into training which comprised of data from the year 2015 to 2018, validation comprising of data from 2018 to September 2019 and test data sets for the last three months of the year 2019. The different divisions of the data sets was tried and the corresponding results have been discussed in further sections. See Section 5.

Deep neural networks

The deep neural network architecture for the centralized approach contains a lambda layer to use the TensorFlow functions, dense layer with 64 units, a dropout layer to prevent the model from overfitting, dense layer with 96 units and then a reshape layer to shape the output as required. This architecture gave good results for the centralized model after trying out several different architectures. The figure 21 shows the architecture of the DNN model used as one of the centralized model.

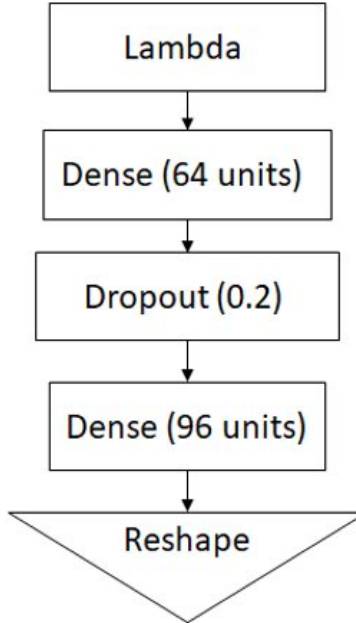


Figure 21: DNN model architecture

Another DNN Model implemented was built using 2 layers with 150 and 50 units respectively with a dropout of 0.2. The activation function used was Rectified Linear Unit (ReLu) and the model was trained for 50 epochs with early stopping being used on the basis of val loss.

Long Short Term Memory Networks (LSTM)

The Long Short Term Memory network architecture for the centralized approach contains a LSTM layer with 50 units, two dropout layer to avoid overfitting, two dense layer with one having 100 and 96 units in each. Final a reshape layer to shape the output as required. The figure 22 shows the architecture of the LSTM model used as one of the centralized model.

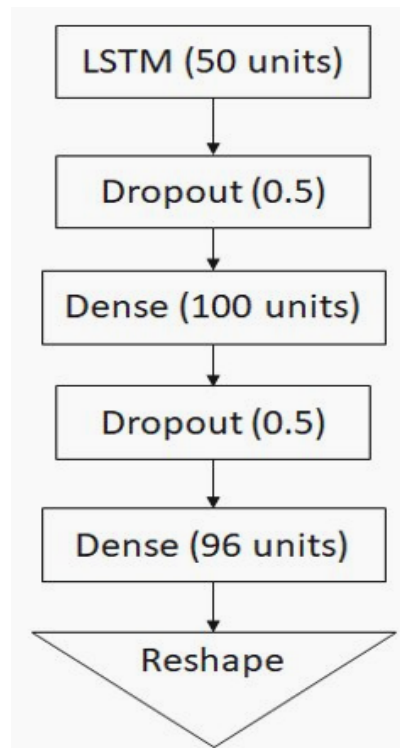


Figure 22: LSTM model architecture

Attention Model

The Attention Model was implemented based on the idea of Attention proposed by Bahdanau [37]. A pre-attention Bi-directional LSTM was first defined followed by performing One step of Attention [42] which outputs the context vector as a dot product of the attention weights and the hidden states of the Bi-directional LSTM. The post-attention LSTM cell was then applied to the context vector and dense layer applied to the post-attention LSTM's hidden state output. The output obtained was appended to the list of outputs. Finally, the model instance was created to return the list of outputs.

The attention model was tried for 24 time steps to make prediction for next one hour of PM10 values with 10 and 25 features from the following - NO2, NOX as NO2, PM10, PM2.5, Day sin, Day cos, Year sin, Year cos, Friday, Monday, Saturday, Sunday, Thursday, Tuesday, Wednesday, dawn, morning, noon, afternoon, evening, midnight, Air Pressure, Temperature Precipitation Amount, Relative Humidity.

4.2.3 Federated Learning Model

Federated model was implemented on the LSTM model as it had the better results when compared to the other models. The federated model was implemented over four clients. The four clients were the four stations, they are: Stockholm E4/E20 Lilla Essingen (18644), Stockholm Sveavagen 59 Gata (8779), Stockholm Hornsgatan 108 Gata (8780) and Stockholm Torkel Knutssonsgatan (8781). The federated learning was implemented using tensorflow tutorial. The figure 23 shows the architecture of the federated model implemented on the four different stations.

Federated learning requires a federated data set, i.e., a collection of data from multiple users. Federated data is typically non-i.i.d., users typically have different distributions of data depending on usage patterns.[43] Some clients may have fewer training examples on device, suffering from data paucity locally, while some clients might have more than enough training examples.[43]

In order to use any model with TFF, it needs to be wrapped in an instance of the `tff.learning.Model` interface, which exposes methods to stamp the model's forward pass, metadata properties, etc., similarly to Keras, but also introduces additional elements, such as ways to control the process of

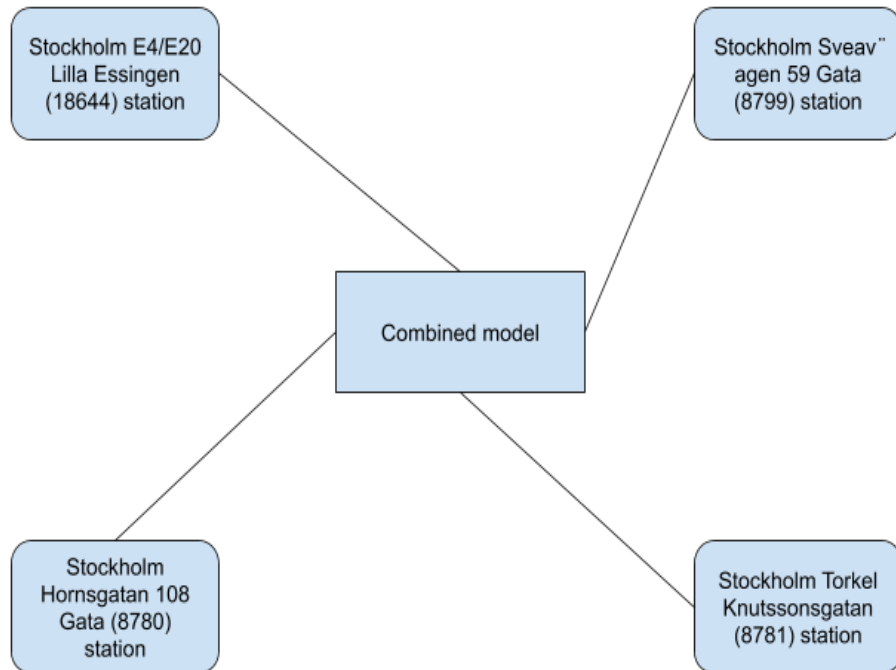


Figure 23: Federated model architecture

computing federated metrics.[43] This can be done using TFF wrap by in-

voking `tff.learning.from_keras_model`.

Algorithm 1: FederatedAveraging. The K clients are indexed by k ; B is the local minibatch size, E is the number of local epochs, and η is the learning rate.[44]

Server executes:

initialize w_0 ;

for each round $t = 1, 2, \dots$ **do**

$m \leftarrow \max(C \cdot K, 1)$;

$S_t \leftarrow$ (random set of m clients);

for each client $k \in S_t$ **do**

$$w_{t+1}^k \leftarrow \text{ClientUpdate}(k, w_t) \quad (4)$$

$$w_{t+1} \leftarrow \sum_{k=1}^K \frac{n_k}{n} w_{t+1}^k \quad (5)$$

end

end

ClientUpdate(k,w): //Run on client k

$B \leftarrow$ (*split* P_k into batches of size B);

for each local epoch i from 1 to E **do**

for batch $b \in B$ **do**

$$w \leftarrow w - \eta \Delta l(w; b) \quad (6)$$

end

return w to server

end

5 Results and Discussion

In this project, the historical air quality data(hourly aggregated) and weather data for different stations were collected from the SMHI website. The meteorological features used for the analysis include Air temperature, Precipitation, Relative humidity, Air pressure, Wind direction, and Wind speed. The training data comprises of years 2015-2018, validation data comprises of months Jan-Sep of year 2019 and test data comprises of months Oct-Dec of year 2019.

The goal was to develop a centralized model that fits best for the data and then to use the optimal centralized model for building the federated learning model. Several pre-processing steps were performed on the data before applying the model. All the data values are normalized and KNN imputation technique was used for the missing values. Several time-features were also added to the data like "Hour of the day" and "Weekday/Weekend" which have shown to improve the performance of the model. Symmetric Mean Absolute Percentage Error (SMAPE) Mean Absolute Error(MAE) were used as performance metrics. Two different model approaches for the centralized model, LSTM and DNN models, were used for forecasting the air pollution. Three prediction intervals - 24-hour prediction, 6 hour prediction, and 1-hour prediction have been experimented and the results were recorded.

LSTM model

Different LSTM models were experimented for the air quality prediction, the best model has the following The network architecture for the centralized approach contains a LSTM layer with 50 units, two dropout layers to avoid over-fitting, two dense layers with 100 and 96 units in each layer and final reshape layer to shape the output as required. All the negative values were kept unchanged and the missing values were filled with KNN imputation method with the K=2. The parameters for this model include

epochs = 30
Batch size = 80
Optimizer = Adam
Input window length = 24
Output features = 4 (PM 10, PM 2.5, NO₂, NO_x as NO₂)
Output window length = 1,6,24

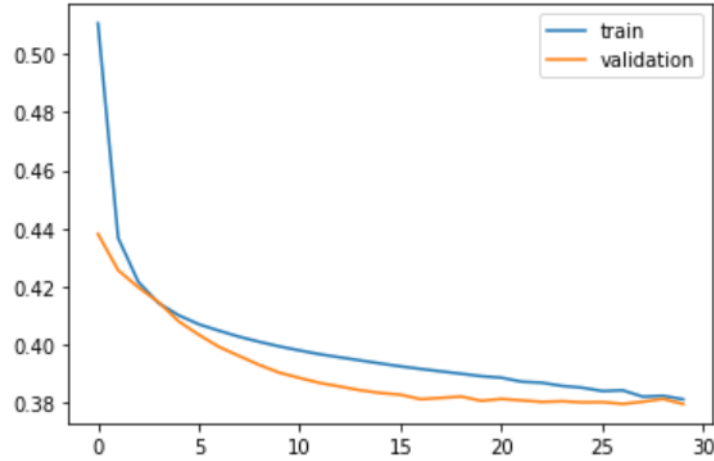


Figure 24: Training Vs Validation loss for LSTM

SMAPE scores were calculated for the normalized predicted values. The average SMAPE value for the best LSTM all the 4 pollutants combined was 0.441 (for 6 hour prediction)

DNN model

The network architecture for this model contains a dense layer with 64 units, a dropout layer to prevent the model from over-fitting, dense layer with 96 units and then a reshape layer to shape the output as required. All the negative values were replaced by 0 and the missing values were filled with KNN imputation method with the $K=4$. The parameters for this model include

epochs = 50

Batch size = 80

Optimizer = Adam

Input window length = 24

Output features = 4 (PM 10, PM 2.5, NO₂, NO_x as NO₂)

Output window length = 1,6,24

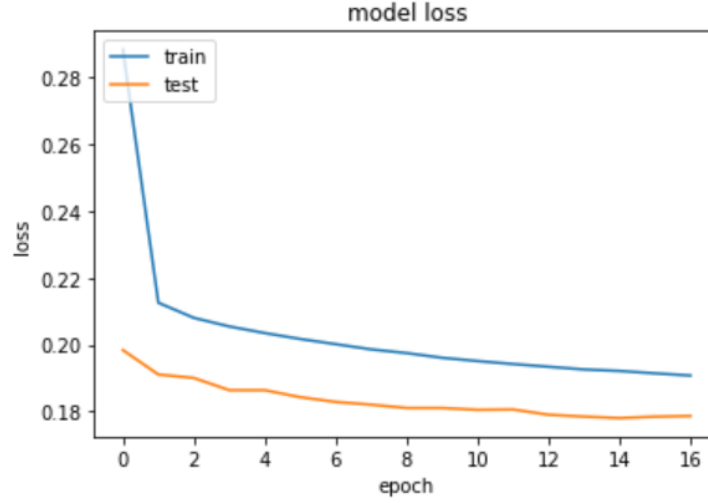


Figure 25: Training Vs Validation loss for DNN

SMAPE scores were calculated for the predicted values. The average SMAPE value for the best LSTM all the 4 pollutants combined was 0.4 (for 6 hour prediction)

LSTM model with Attention

Bidirectional LSTM model with attention based on the idea proposed by Bahdanau [37] was implemented to see if there are changes in the performance of the model. The parameters used for this approach are:

epochs = 50 with EarlyStopping

Batch size = 72

Input window = 24

Output window = 1

Output features = 4 (PM 10, PM 2.5, NO₂, NO_x as NO₂)

The results can be seen in Table 3

Model	Output Window	NO2	NOX as NO2	PM10	PM2.5	AvgScore
LSTM	6-hours	0.3854	0.454	0.3603	0.5645	0.4412
LSTM	1-hour	0.3269	0.3893	0.3188	0.3152	0.3376
LSTM	24-hours	0.5269	0.4397	0.4475	0.466	0.4702
Attention	1-hour	0.249	0.323	0.3487	0.287	0.301
DNN	6-hours	0.417	0.429	0.404	0.357	0.4024
DNN	1-hour	0.335	0.3	0.239	0.252	0.282
DNN	24-hours	0.541	0.523	0.57	0.45	0.5214

Table 3: SMAPE scores of the different models implemented for the centralized model.

Model	Output Window	NO2	NOX as NO2	PM10	PM2.5	AvgScore
LSTM	6-hours	0.204	0.204	0.179	0.341	0.232
LSTM	1-hour	0.208	0.166	0.144	0.219	0.184
LSTM	24-hours	0.503	0.446	0.329	0.565	0.461
DNN	6-hours	0.4	0.385	0.275	0.3885	0.4
DNN	1-hour	0.24	0.26	0.18	0.23	0.22
DNN	24-hours	0.53	0.45	0.36	0.57	0.47

Table 4: MAE scores of the different models implemented for the centralized model.

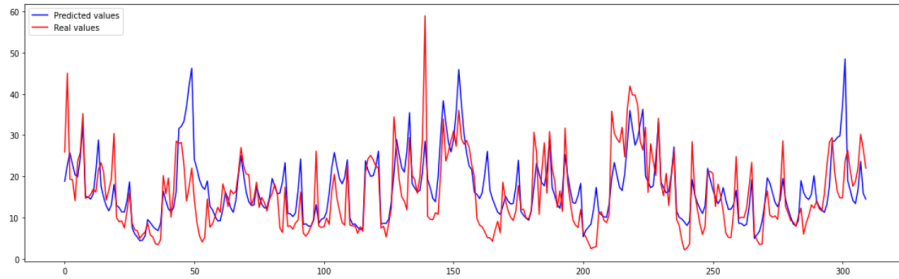


Figure 26: Actual vs Prediction for NO2 using LSTM(6-hours prediction)

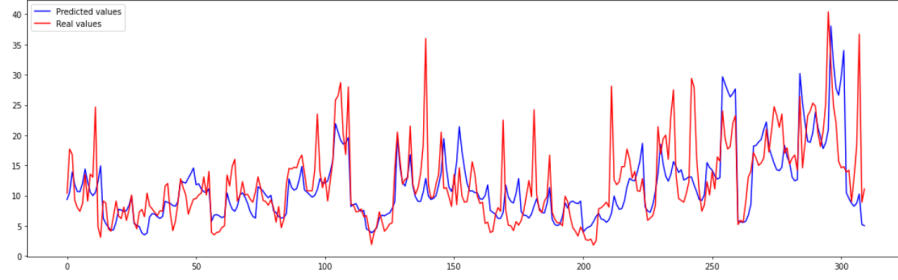


Figure 27: Actual vs Prediction for PM10 using LSTM(6-hours prediction)

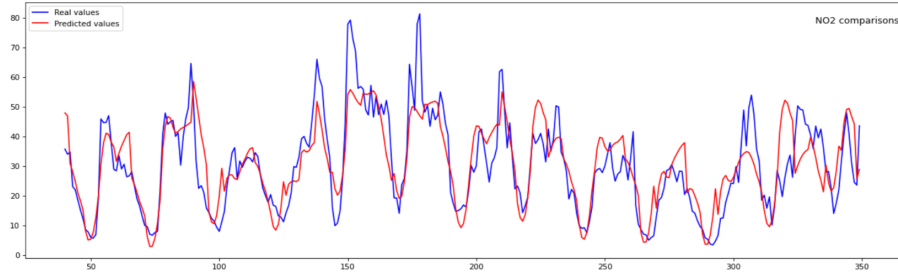


Figure 28: Actual vs Prediction for NO2 using DNN(6-hours prediction)

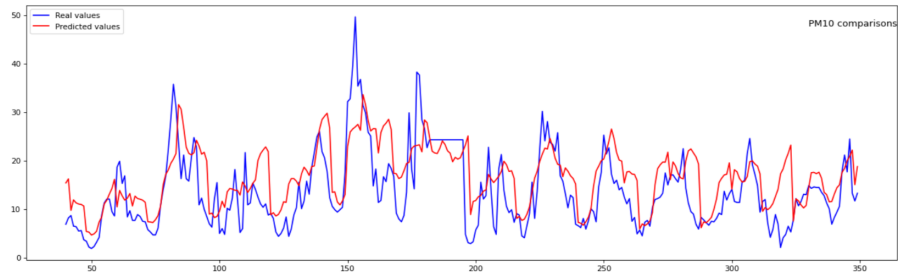


Figure 29: Actual vs Prediction for PM10 using DNN(6-hours prediction)

Federated Learning

The best performing LSTM model was used for creating a federated model. "tensorflow federated" package was used for creating the federated model. The parameters used for this approach are:

epochs = 50
Batch size = 80
Input window = 24
Output features = 4 (PM10, PM2.5, NO2, NOx as NO2)

For this federated learning approach, 4 different stations data have been used as 4 clients and a global model was developed based on these 4 clients. MAE (Mean Absolute Error) was used as a performance metric. The federated model has the same training and test data as the centralized model and the model performance was evaluated for the interval Oct-Dec of year 2019(same as centralized model).

For the global model, the MAE score which was evaluated for all the clients was 0.5334.

The scores for global model evaluated per individual client are given by:

Station id	MAE Score
8779	0.51546055
8780	0.56634694
8781	0.53623414
18644	0.5156667

Table 5: Federated Learning: MAE scores for global model evaluated per individual station

6 Decentralised Air Quality Prediction and Monitoring (DAMP) - The Product

6.1 DAMP Web Application

The results obtained from the models were not easily understandable to users. Going through each timeframe and finding the values could be troublesome to the users. Showing the data and a graphical representation could be useful, but making it more understandable, and an Interactive display might help users better understand the values and compare them. So the best idea was to create a Web App. A simple and easy App where users can select a particular date and time and see the data's visual representation.

The App is simple and easy to use. It is a responsive app that can be accessed easily from mobile phones, tablets, or pc. The app URL directs users to the main page, where users have to choose the date and time. When the user selects the date and time, the App will show the user selected hour prediction and the 24 hours predicted values for the same day.

DAMP Home About

Decentralized Air quality Monitoring and Prediction

Pick a date

< December 2019 >

Su	Mo	Tu	We	Th	Fr	Sa
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31	1	2	3	4
5	6	7	8	9	10	11

01

Get prediction

Copyright © Uppsala Project CS 2020

Figure 30: WebApp Main Page

Once the user selects date and time, the selected hour values and the 24-hour values are visualized using two different graphs. A solid gauge is used to represent the current hour air quality values, and a basic line chart is used to show the 24 hours of air quality predictions. The solid gauge represents the maximum and the minimum values of the specific air pollutants. Where it also displays if the current air pollutant values are healthy or unhealthy. This helps the users to understand if it is safe to outside or not. The standard values are the European standard air quality values.[45]

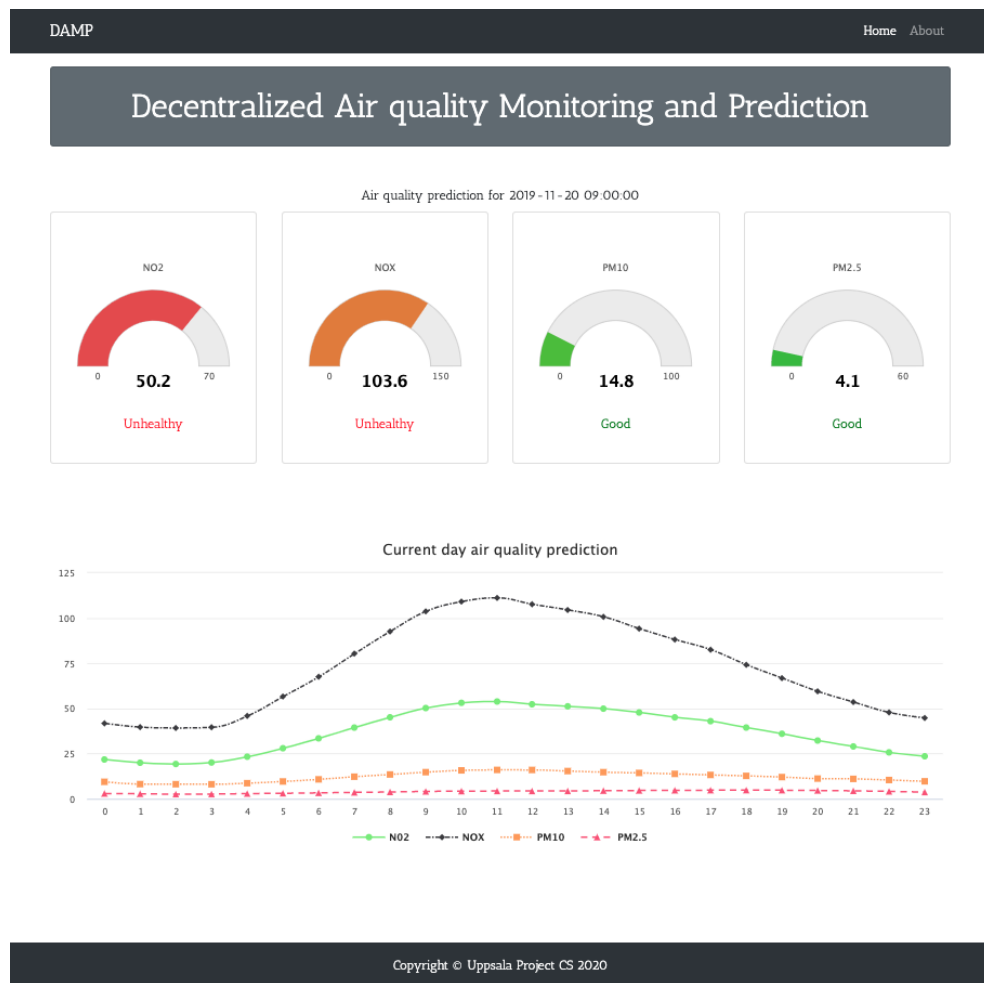


Figure 31: Airquality values

6.1.1 Technologies and Libraries

- **Flask**

Flask is a micro web framework created by **Armin Ronacher** written in Python. The "micro" means that Flask aims to keep the core simple but extensible. Since the Flask is written in Python, the Flask packages are easily available, which can be installed from PyPI, an official third-party software repository for Python. Flask is based on the Werkzeug WSGI toolkit and Jinja2 template engine.

Flask is simple, easy, and flexible. Flask is more popular among the users as it allows users to make their decisions on using the database, engines, and interacting with them.

Flask is a backend framework. For this project, the backend needed to be an API that reads and writes data files and returns JSON objects. The database was not required for this project. The most convenient and more straightforward way to display the air pollutants' values to the customer was to build a WebApp using Flask.

- **jQuery**

jQuery is a JavaScript library created by **John Resig**. It is a lightweight, fast, and powerful library that makes it easier to use JavaScript on a webpage. jQuery library contains features like HTML/DOM manipulation, CSS manipulation, AJAX, effects, and animations and has plugins for almost all kinds.

Due to its compatibility with most browsers, it is popular among web developers.

- **Bootstrap**

Bootstrap was developed by the two developers **Mark Otto** and **Jacob Thornton**, at Twitter. It is a popular open-source front-end framework used for creating modern web apps and websites. The reason for its popularity is due to its responsiveness, customizable bootstrap, simple integration, grid, pre-styled components, and consistency. It is easy to set up and master. In this WebApp, it is used mainly for the responsive CSS and grid.

- **Highcharts**

Highcharts is a pure Javascript library created by Highsoft and is used for making charts on webpages. Due to its compatibility with all modern mobile and desktop browsers, it has gained popularity. The source codes are free to download and make your edits.

In this WebApp, only two types of charts are used. The basic line charts showing the trends of the dataset. This line chart shows the 24-hour air quality values. The other is the solid gauge with dynamic data. Four different charts are used for displaying the current hour air quality data. It also shows if the values of the pollutants are good or Unhealthy. This helps the users to understand if it is better to go outside or not.

7 Conclusions

Air Pollution is a significant risk factor for several pollution related diseases including respiratory infections, heart diseases, lung cancer etc. The affect of poor air on a human body is far reaching. It is estimated that air pollution kills around seven million people worldwide each year. This make it very important for the air to be monitored continuously. Monitoring the level of pollutants' concentration in the air help us guard against extreme events by alerting people and initiating action. By way of this project, we intend to monitor the concentration of pollutants (PM10, PM2.5, NO2 and NOX as NO2) in the air of Stockholm. To predict the values of the pollutants, centralized models and federated learning models are used and the performance is compared.

The dataset (air pollutants and meteorological data) that was obtained from SMHI for the years 2015-2019, had a lot of missing data and data that had to be cleaned. The group did a lot of literature survey on the best method to impute the missing values. Some of the imputation techniques tried were KNN, Multiple Imputation by Chained Equations, Mean/Median etc. Based on performance, it was decided that KNN imputation would be used since it gave the best performance. Although the missing values were filled, the dataset still had to be deal with negative values which were not possible since the air cannot have negative concentrations of a pollutant. The negative values were replaced by 0 since the range was not too low and hence it was deemed as a calibration error.

In addition to the above-mentioned features, various date time features like Day Sin, Day cos, Year sin, Year cos, weekday, time of the day were used. The entire dataset was split into train, validation, and testing datasets. The dataset was standardized before forwarding it to the model to train.

An LSTM and a DNN model were trained on the dataset and their performance was compared. The prediction is done for 24 hours, 6 hours and 1 hour and their performance is compared. Table 3 and Table 4 demonstrates the comparison of the performance. It can be inferred from the tables that the lesser the input window and the prediction window, the better is the prediction.

The LSTM model was further used to perform federated learning. The package used for creating the federated model is 'Tensorflow Federated'. The project uses horizontal federated learning where all workers have the same features but different samples. In this case, the different samples are from different stations. To evaluate the performance, Mean Absolute Error(MAE) was used. MAE Scores for different stations can be seen in table 5. After referring the result tables 4 and 5, it can be seen that there is not much difference between the performances based on the MAE scores. Though, it can be inferred that federated learning model is an answer to critical issues such as data security, data privacy and illegal access of data.

8 Future Work

There is some work that could not be done due to time and resource constraints. It would be great if we could do some real time predictions and deploy our model to fetch real time data. Following are the points that can be done further in this project:

- The Web-application currently we have, works with already predicted values. We could integrate the web app with the backend. This could help us in making live predictions as well
- We have developed an attention model but due to time constraints, we couldn't visualize the attention weights for the input attributes. This could help us to understand which attribute has more influence on the output.
- If there was more time, we could explore more about the parameters of the attention model.

Acronyms

DAMP Decentralized Air Quality Monitoring and Prediction. 22

DNN Deep neural network. 13, 27, 30, 34

FL Federated Learning. 17

K-NN K-Nearest Neighbour. 11

LOF Local Outlier Factor. 12

LSTM Long Short Term Memory Networks. 15, 27, 30, 34

MAE Mean Absolute Error. 18, 34

MCD Minimum Co-variance Determinant. 12

RNN Recurrent Neural Networks. 14, 15, 20

SMAPE Symmetric Mean Absolute Percentage Error. 18, 34

SMHI Sweden’s Meteorological and Hydrological Institute. 1, 23, 24

References

- [1] B. Columbia. (). “Environmental protection and sustainability: Air, land and water,” [Online]. Available: <https://www2.gov.bc.ca/gov/content/environment/air-land-water/air>.
- [2] M. Waskom and the seaborn development team, *Mwaskom/seaborn*, version latest, Sep. 2020. DOI: 10.5281/zenodo.592845. [Online]. Available: <https://doi.org/10.5281/zenodo.592845>.
- [3] A. Bilogur, “Missingno: A missing data visualization suite,” *Journal of Open Source Software*, vol. 3, no. 22, p. 547, 2018. DOI: 10.21105/joss.00547. [Online]. Available: <https://doi.org/10.21105/joss.00547>.

-
- [4] J. D. Hunter, “Matplotlib: A 2d graphics environment,” *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007. DOI: 10.1109/MCSE.2007.55.
 - [5] Z. Gu, R. Eils, and M. Schlesner, “Complex heatmaps reveal patterns and correlations in multidimensional genomic data,” *Bioinformatics*, vol. 32, no. 18, pp. 2847–2849, May 2016, ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btw313. eprint: <https://academic.oup.com/bioinformatics/article-pdf/32/18/2847/24406805/btw313.pdf>. [Online]. Available: <https://doi.org/10.1093/bioinformatics/btw313>.
 - [6] M. Ebrahim, *Python correlation matrix tutorial*, Nov. 2020. [Online]. Available: <https://likegeeks.com/python-correlation-matrix/>.
 - [7] ResidentMario, *Residentmario/missingno*. [Online]. Available: <https://github.com/ResidentMario/missingno>.
 - [8] *Seaborn.violinplot*. [Online]. Available: <https://seaborn.pydata.org/generated/seaborn.violinplot.html>.
 - [9] D. Nelson, *Seaborn violin plot - tutorial and examples*. [Online]. Available: <https://stackabuse.com/seaborn-violin-plot-tutorial-and-examples/>.
 - [10] E. Lewinson, *Violin plots explained*, Feb. 2020. [Online]. Available: <https://towardsdatascience.com/violin-plots-explained-fb1d115e023d>.
 - [11] D. A. Keim, M. C. Hao, U. Dayal, H. Janetzko, and P. Bak, “Generalized scatter plots,” *Information Visualization*, vol. 9, no. 4, pp. 301–311, Dec. 2010, ISSN: 1473-8716. DOI: 10.1057/ivs.2009.34. [Online]. Available: <https://doi.org/10.1057/ivs.2009.34>.
 - [12] Cmdline, *How to highlight data points with colors and text in python?* Jan. 2020. [Online]. Available: <https://cmdlinetips.com/2019/11/how-to-highlight-data-points-with-colors-and-text-in-python/>.
 - [13] *Seaborn.pairplot*. [Online]. Available: <https://seaborn.pydata.org/generated/seaborn.pairplot.html>.
 - [14] *Time series analysis: Identifying ar and ma using acf and pacf plots*. [Online]. Available: <https://towardsdatascience.com/tagged/autocorrelation>.

-
- [15] A. Anderson and A. the Book Author Alan Anderson, *Autocorrelation plots: Graphical technique for statistical data*. [Online]. Available: <https://www.dummies.com/programming/big-data/data-science/autocorrelation-plots-graphical-technique-for-statistical-data/>.
 - [16] *Methods for imputation of missing values in air quality data sets*. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1352231004001815>.
 - [17] *Mice*. [Online]. Available: <https://stats.stackexchange.com/questions/421545/multiple-imputation-by-chained-equations-mice-explained>.
 - [18] *Multiple imputation by chained equations: What is it and how does it work?* [Online]. Available: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2727238/>.
 - [19] P. R. M. Hubert M. Debruyne, “Minimum covariance determinant and extensions,” *Wiley Interdiscip. Rev. Comput. Stat.*, no. 10 (3) Article e1421, 2018.
 - [20] C. Chatfield, *Time series Forecasting*. Chapman Hallmark/crc, 2000.
 - [21] G. Bontempi, S. Ben Taieb, and Y.-A. Le Borgne, “Machine learning strategies for time series forecasting,” in *Business Intelligence: Second European Summer School, eBISS 2012, Brussels, Belgium, July 15-21, 2012, Tutorial Lectures*, M.-A. Aufaure and E. Zimányi, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 62–77, ISBN: 978-3-642-36318-4. DOI: 10.1007/978-3-642-36318-4_3. [Online]. Available: https://doi.org/10.1007/978-3-642-36318-4_3.
 - [22] S. Hosein and P. Hosein, “Load forecasting using deep neural networks,” in *2017 IEEE Power Energy Society Innovative Smart Grid Technologies Conference (ISGT)*, 2017, pp. 1–5. DOI: 10.1109/ISGT.2017.8085971.
 - [23] J. Schmidhuber, “Deep learning in neural networks: An overview,” *Neural Networks*, vol. 61, pp. 85–117, Jan. 2015, ISSN: 0893-6080. DOI: 10.1016/j.neunet.2014.09.003. [Online]. Available: <http://dx.doi.org/10.1016/j.neunet.2014.09.003>.
 - [24] B. Yoshua, *Learning Deep Architectures for AI*. Foundations and Trends in Machine Learning, 2009, vol. 2, pp. 1–127. DOI: 10.1561/22000000006.

-
- [25] G. D. Robert DiPietro, “Deep learning: Rnns and lstm,” *Handbook of Medical Image Computing and Computer Assisted Intervention*, no. B978-0-12-816176-0.00026-0, 2019.
- [26] C. Olah. (). “Understanding lstm networks,” [Online]. Available: <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>.
- [27] T. Shen, T. Zhou, G. Long, J. Jiang, S. Pan, and C. Zhang, *Disan: Directional self-attention network for rnn/cnn-free language understanding*, 2017. arXiv: 1709.04696 [cs.CL].
- [28] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, “Federated learning: Challenges, methods, and future directions,” *CoRR*, vol. abs/1908.07873, 2019. arXiv: 1908.07873. [Online]. Available: <http://arxiv.org/abs/1908.07873>.
- [29] B. McMahan and D. Ramage. (). “Federated learning: Collaborative machine learning without centralized training data,” [Online]. Available: <https://ai.googleblog.com/2017/04/federated-learning-collaborative.html>.
- [30] Wikipedia. (). “Mean absolute error,” [Online]. Available: https://en.wikipedia.org/wiki/Mean_absolute_error.
- [31] —, (). “Symmetric mean absolute percentage error,” [Online]. Available: https://en.wikipedia.org/wiki/Symmetric_mean_absolute_percentage_error.
- [32] S. Edgerton, X. Bian, J. Doran, J. Fast, J. Hubbe, E. Malone, W. Shaw, C. Whiteman, S. Zhong, J. Arriaga, E. Ortiz, M. Ruiz, G. Sosa Iglesias, E. Vega, T. Limon, F. Guzman, J. Archuleta, J. Bossert, S. Elliot, and R. Petty, “Particulate air pollution in mexico city: A collaborative research project,” *Journal of the Air Waste Management Association*, vol. 49, pp. 1221–1229, Oct. 1999. DOI: 10.1080/10473289.1999.10463915.
- [33] Z. Luo, J. Huang, K. Hu, X. Li, and P. Zhang, “Accuair: Winning solution to air quality prediction for kdd cup 2018,” Jul. 2019, pp. 1842–1850, ISBN: 978-1-4503-6201-6. DOI: 10.1145/3292500.3330787.
- [34] B. Freeman, G. Taylor, B. Gharabaghi, and J. Thé, “Forecasting air quality time series using deep learning,” *Journal of the Air Waste Management Association*, vol. 68, Nov. 2017. DOI: 10.1080/10962247.2018.1459956.

-
- [35] H. G. van Zoest VM Stein A, “Outlier detection in urban air quality sensor networks,” *Water Air Soil Pollut.*, vol. 229, no. 4, p. 111, 2018. DOI: 10.1007/s11270-018-3756-7. [Online]. Available: doi:10.1007/s11270-018-3756-7.
 - [36] Z. Luo, J. Huang, K. Hu, X. Li, and P. Zhang, “Accuair: Winning solution to air quality prediction for kdd cup 2018,” in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery Data Mining*, ser. KDD ’19, Anchorage, AK, USA: Association for Computing Machinery, 2019, pp. 1842–1850, ISBN: 9781450362016. DOI: 10.1145/3292500.3330787. [Online]. Available: <https://doi.org/10.1145/3292500.3330787>.
 - [37] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, “Learning phrase representations using RNN encoder–decoder for statistical machine translation,” in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1724–1734. DOI: 10.3115/v1/D14-1179. [Online]. Available: <https://www.aclweb.org/anthology/D14-1179>.
 - [38] Q. Yang, Y. Liu, T. Chen, and Y. Tong, “Federated machine learning: Concept and applications,” *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 2, Jan. 2019, ISSN: 2157-6904. DOI: 10.1145/3298981. [Online]. Available: <https://doi-org.ezproxy.its.uu.se/10.1145/3298981>.
 - [39] S. Meteorological and H. I. (SMHI). (). “Historical data for air quality,” [Online]. Available: <https://datavardluft.smhi.se/portal/concentrations-in-air..>
 - [40] —, (). “Weather data,” [Online]. Available: <https://www.smhi.se/data/meteorologi/ladda-ner-meteorologiska-observationer/..>
 - [41] C. M. González-Duque, J. Cortés-Araujo, and B. H. Aristizábal-Zuluaga, “Influence of meteorology and source variation on airborne Pm10 levels in a high relief tropical Andean city,” en, *Revista Facultad de Ingeniería Universidad de Antioquia*, pp. 200–212, Mar. 2015, ISSN: 0120-6230. [Online]. Available: http://www.scielo.org.co/scielo.php?script=sci_arttext&pid=S0120-62302015000100018&nrm=iso.

-
- [42] A. Sinha and D. Muddappa. (). “Solve air: Air pollution forecasting using deep attentive sequence learning,” [Online]. Available: <https://github.com/abinashsinha330/Air-Pollution-Forecasting-using-Machine-Learning-APF-Model>.
- [43] M. R. Zachary Garrett. (). “Federated learning for image classification,” [Online]. Available: https://github.com/tensorflow/federated/blob/master/docs/tutorials/federated_learning_for_image_classification.ipynb.
- [44] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, “Communication-efficient learning of deep networks from decentralized data,” in *Artificial Intelligence and Statistics*, PMLR, 2017, pp. 1273–1282.
- [45] E. E. Agency. (). “Air quality standards under the air quality directive, and who air quality guidelines,” [Online]. Available: <https://www.eea.europa.eu/themes/data-and-maps/figures/air-quality-standards-under-the>.

A Appendix A: Instructions to use the Web Application

Flask Web Application developed to demonstrate the air quality prediction

A.1 Step I

Users have to go through the app URL, and they are directed towards the main app page.

A.2 Step II

Users need to select the date and time for which they want to see Air pollutants' values using the date picker.

A.3 Step III

Once the users select the date and time, they have to hit the get predictions button.

A.4 Step IV

The get prediction directs the users to the predicted values. They are represented by two different graphs. The upper graph is the gauge meter showing the selected hour values for NO₂, NO_X, PM₁₀ and PM_{2.5}. The lower line bar shows the 24-hour air quality values for all the four pollutants.